universität wien

Bachelorarbeit

Titel der Bachelorarbeit

„Darstellung von Kundenverhalten mittels einer Android App und Daten aus Perceptrons“

Verfasser
Nicolas Mattersdorfer 11807110

angestrebter akademischer Grad

Bachelor of Science (BSc)

Wien, 2023

Studienkennzahl lt. Studienblatt:	A 033 526
Fachrichtung:	Wirtschaftsinformatik
Betreuerin / Betreuer:	Dipl.-Ing. Marian Lux

Eidesstattliche Erklärung

Ich Nicolas Mattersdorfer Student an der Universität Wien (Matrikelnummer: 11807110) erkläre eidesstattlich, dass ich die Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und alle aus ungedruckten Quellen, gedruckter Literatur oder aus dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte gemäß den Richtlinien wissenschaftlicher Arbeiten zitiert, durch Fußnoten gekennzeichnet bzw. mit genauer Quellenangabe kenntlich gemacht habe. Diese schriftliche Arbeit wurde noch an keiner anderen Stelle vorgelegt.

05.05.2023 Nicolas Mattersdorfer

Datum Unterschrift Studierender

Vorwort

Diese Bachelorarbeit wurde an der Universität Wien im Bereich Wirtschaftsinformatik geschrieben und beschäftigt sich mit der Visualisierung von Process Mining und Perceptron in einem Android Projekt. Für das Projekt wurde ein trainiertes Model verwendet und aus den Ergebnissen und erhaltenen Daten wurde eine Android-Applikation entwickelt, welche diese aufbereitet und dem User in verschiedenen Kategorien und Übersichten zur Verfügung stellt. Durch diese Kategorien und Aufbereitung soll eine bessere Lesbarkeit für den User generiert werden. Zusätzlich zu dem praktischen Projekt soll die Bachelorarbeit einen kurzen theoretischen Hintergrund zu Process Mining im Allgemeinen geben.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

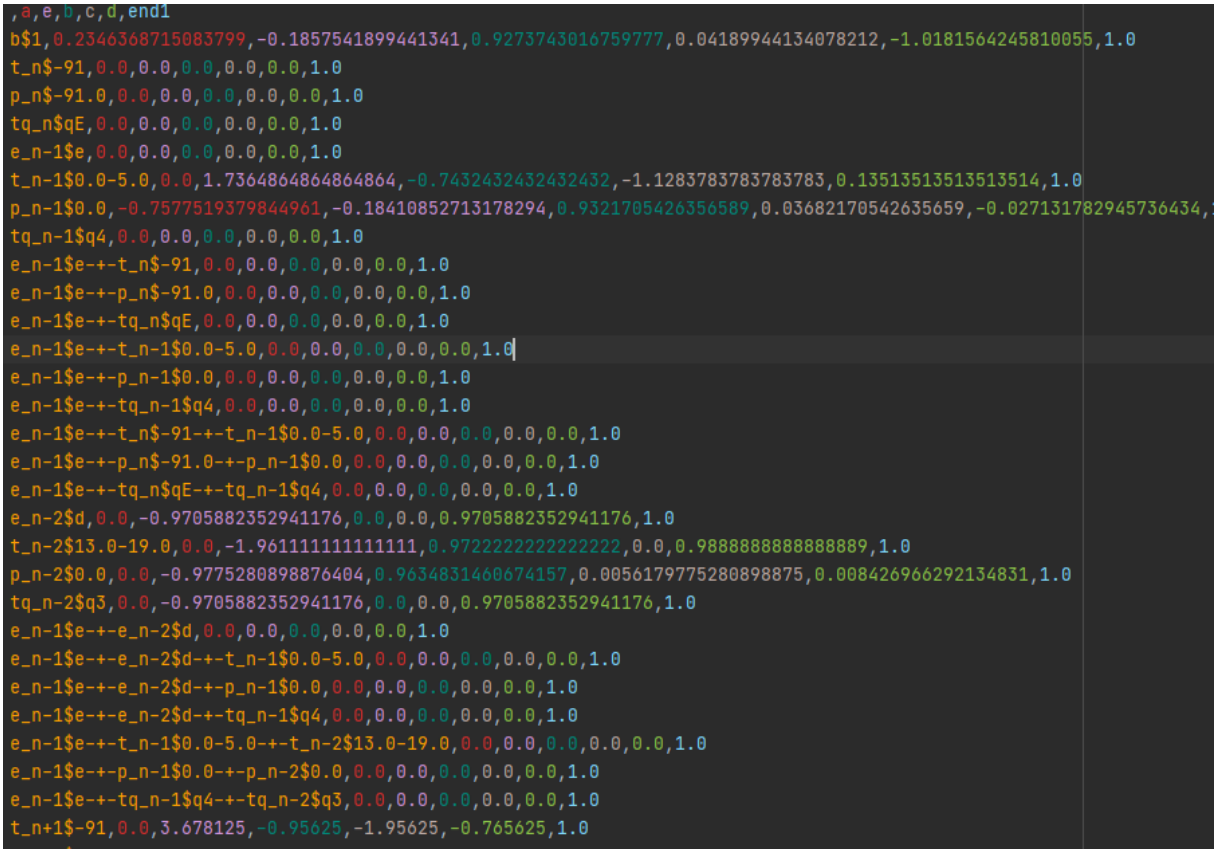
In der modernen Zeit wird die Optimierung von Daten und Kundenverhalten immer wichtiger, um Produkte und Dienstleistungen gewinnbringend zu vermarkten. Viele dieser neuen Applikationen oder Produkte werden so konzipiert, dass sie über eine lange Laufzeit optimiert werden. Durch diese Optimierung ist es möglich, den Informationsgehalt von Daten zu erhöhen.

Im Rahmen dieser Bachelorarbeit wird zuerst ein kleiner theoretischer und praktischer Hintergrund gegeben und anschließend die Struktur und Umsetzung der Theorie in einer Android-Applikation wiedergegeben. Für den Theorie-Aspekt wird auf Process Mining in Kombination mit Perceptrons eingegangen und ein grober Überblick gegeben. Diese gewonnenen Informationen und Theorie-Aspekte helfen dabei, die Struktur und den Aufbau der Android Applikation zu bilden und anschließend die notwendigen Daten zu verwenden und anzureichern. Somit beschäftigt sich der praktische Teil mit der Umsetzung der Theorie-Aspekte auf echte Datensätze.

1.1. Ausgangslage

Es gibt schon eine Applikation für IOS, die einige der angestrebten Funktionen beinhaltet, aber nicht alle, die implementiert werden sollen. Die alte Applikation sollte vom Konzept angepasst und in Android aufgebaut werden. Hierzu ist eine Menge an Daten und Strukturen notwendig. Die Grunddaten wurden am Start der Bachelorarbeit zur Verfügung gestellt und mussten verwendet werden. Diese Daten bilden ein fertig trainiertes Perceptron und die Werte und Kategorien sind die Grunddaten für die Struktur der Android Applikation. Diese Daten wurden in einem CSV-File bereitgestellt und sollten auch so verwendet werden, dass in Zukunft neue Daten durch das Programm erkannt und verarbeitet werden können.

Als Beispiel für diese Strukturen und das File dient dieser CSV-Ausschnitt.



```
,a,e,b,c,d,end1
b$1,0.2346368715083799,-0.1857541899441341,0.9273743016759777,0.04189944134078212,-1.0181564245810055,1.0
t_n$-91,0.0,0.0,0.0,0.0,0.0,1.0
p_n$-91.0,0.0,0.0,0.0,0.0,0.0,1.0
tq_n$qE,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e,0.0,0.0,0.0,0.0,0.0,1.0
t_n-1$0.0-5.0,0.0,1.7364864864864864,-0.7432432432432432,-1.1283783783783783,0.13513513513513514,1.0
p_n-1$0.0,-0.7577519379844961,-0.18410852713178294,0.9321705426356589,0.03682170542635659,-0.027131782945736434,
tq_n-1$q4,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-t_n$-91,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-p_n$-91.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-tq_n$qE,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-t_n-1$0.0-5.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-p_n-1$0.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-tq_n-1$q4,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-t_n$-91-+-t_n-1$0.0-5.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-p_n$-91.0-+-p_n-1$0.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-tq_n$qE-+-tq_n-1$q4,0.0,0.0,0.0,0.0,0.0,1.0
e_n-2$d,0.0,-0.9705882352941176,0.0,0.0,0.9705882352941176,1.0
t_n-2$13.0-19.0,0.0,-1.961111111111111,0.9722222222222222,0.0,0.9888888888888889,1.0
p_n-2$0.0,0.0,-0.9775280898876404,0.9634831460674157,0.0056179775280898875,0.008426966292134831,1.0
tq_n-2$q3,0.0,-0.9705882352941176,0.0,0.0,0.9705882352941176,1.0
e_n-1$e-+-e_n-2$d,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-e_n-2$d-+-t_n-1$0.0-5.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-e_n-2$d-+-p_n-1$0.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-e_n-2$d-+-tq_n-1$q4,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-t_n-1$0.0-5.0-+-t_n-2$13.0-19.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-p_n-1$0.0-+-p_n-2$0.0,0.0,0.0,0.0,0.0,0.0,1.0
e_n-1$e-+-tq_n-1$q4-+-tq_n-2$q3,0.0,0.0,0.0,0.0,0.0,1.0
t_n+1$-91,0.0,3.678125,-0.95625,-1.95625,-0.765625,1.0
```

Abbildung 1: CSV-Grunddaten

Hier ist ersichtlich, dass die Datei in folgender Reihenfolge aufgebaut ist. Die erste Reihe bildet die Knotenpunkte am Perceptron ab. Diese haben dann in den folgenden Reihen immer ein „Event“ plus einen dazugehörigen Wert. Hierzu kann man sagen, dass die Events in 4 erkennbare „Identifier“ unterteilbar sind, welche wieder in Unterkategorien unterteilt werden.

Diese Identifier sind:

T_N$ - Temperaturwerte, die aufgetreten sind.

P_N$ - Wetterbedingungen, ob es geregnet hat oder nicht.

TQ_N$ - Tageszeit, wann das Event stattgefunden hat.

E_N$ - Aktivität, die vorrangegangen oder nachfolgend ist.

Diese Werte können auch variieren, wie „T_N-1$“ wäre die Temperatur des vorherigen Events und „T_N+1$“ wäre die Temperatur des nachfolgenden Events. Diese können bis „+2“ und „-2“ gehen und bilden somit die Grundstruktur, die für den Aufbau der Daten in der Applikation essenziell sind.

1.2. Theorie zu Perceptron und Process Mining

Doch warum sind diese Werte, die im letzten Kapitel genannt wurden, überhaupt relevant und wie sind sie durch das Perceptron entstanden? Was ist ein Perceptron und was ist Process Mining? Diese Fragen werden im folgenden Kapitel behandelt und genauer auf die Geschichte der Entstehung und Verwendung eingegangen.

1.2.1. Perceptron

Ein Perceptron ist ein Teil des großen Gebiets des Machine Learning. Grundlegend kann man zwischen zwei verschiedenen Methoden im Machine Learning unterscheiden (vgl. Trask, 2019):

<u>Überwachung</u>
Man kann eine Maschine überwachen und seine Wünsche dementsprechend anpassen, um einen richtigen Output zu erhalten.

<u>Unüberwacht</u>
Man überwacht nicht und die Daten werden von der Maschine gebündelt und Kategorien zugewiesen, welche aber oft nicht die gewünschten sind, die entstehen sollten.

Das Konzept eines Perceptron entstand in dem Jahr 1943 und wurde von W. McCulloch und Walter Pitts entwickelt. Die Idee hinter dem Perceptron war es, eine Maschine unter Beaufsichtigung lernen zu lassen und somit die ganze Maschine zu trainieren. Somit gehört ein Perceptron meist zu dem überwachten Teil des Machine Learning. In den Anfängen war es nicht vorgesehen, dass ein Perceptron ein wirklicher Teil des Programmcodes wird, sondern eine wirkliche reale Maschine. In den meisten Fällen ist es aber in der heutigen Zeit ein Computerprogramm und nicht eine reale Maschine. (vgl. Sabry, 2023)

Die klassische Funktionsweise eines Perceptron ist in der Regel sehr simpel. Es gibt einen binären Wert, der durch eine Klassifizierung läuft und danach einer bestimmten Kategorie zugeteilt wird. Hierbei treten einige Probleme auf, die in den Anfängen nicht bedacht wurden. Da die Klassifizierung abhängig von den Werten und Kategorien ist,

fällt es oft schwer, dass eine hohe Anzahl von verschiedenen Kategorien und Aspekten trainiert werden können. Durch dieses Problem gab es viele Forscher*innen, die sich überlegten, wie eine Verbesserung durchgeführt werden kann. (vgl. Sabry, 2023)

Durch die intensive Forschung entstanden neuronale Netzwerke mit mehreren unterschiedlichen Ebenen, welche auch Layer genannt werden. Dieses Zusammenspielen der einzelnen Layer bildet das Perceptron. Die Daten werden nach jedem Verarbeitungsschritt an die nächste Ebene übergeben, wo sie wieder gebündelt und neuen Kategorien zugeteilt werden. Durch den Layer war es möglich, komplexere Aufgaben und Klassifizierungen durchzuführen. Durch diese Änderungen ist es in der Regel möglich, lineare und nicht-lineare Probleme zu identifizieren und zu lösen. (vgl. Sabry, 2023)

Das folgende Bild soll die grundlegenden Informationen visuell aufzeigen und vereinfachen.

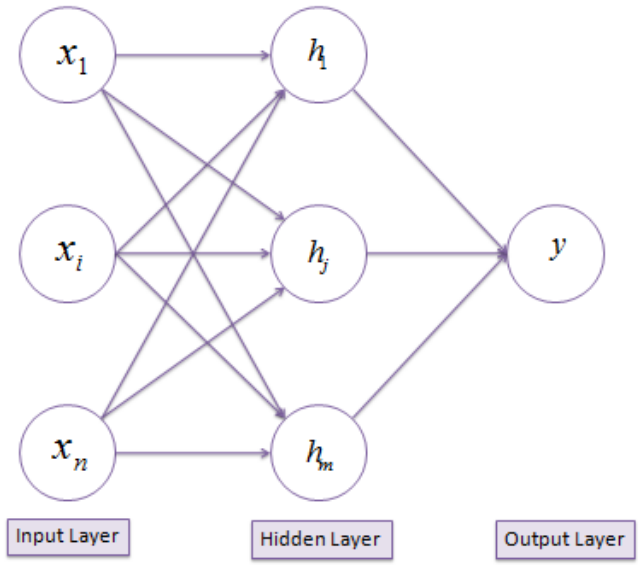


Abbildung 2: Perceptron Struktur

Diese Grafik ist eine simple Darstellung eines mehrdimensionalen Perceptron. Es ist leicht ersichtlich, dass die Werte am Anfang in den Input Layer gehen und durch die Kategorisierung in den Hidden Layer eintreten. Hier werden die Werte immer weiter verändert und angepasst und dann an den Output Layer übergeben. Die Werte werden in der Regel „gewichtet“. Das heißt, dass die Werte mit einer bestimmten Zahl

verändert werden, um am Ende in der Zielfunktion eine größere oder niedrigere Wichtigkeit zu erhalten.

Diese Modelle und Gewichtungen sind in der Regel aber nicht optimal. Dazu gibt es verschiedene Methoden, wie man diese ändern und verbessern kann. Eine dieser Methoden ist „Forward Propagation“ und „Backward Propagation“ beziehungsweise Vorwärtsausbreitung und Rückwärtsausbreitung. Bei dieser Methode werden die Werte zuerst durch das System gewichtet und zum Output Layer transportiert. Am Ende wird überprüft, ob das gewünschte Ergebnis erreicht wurde und sollte dies nicht der Fall sein, wird pro Layer die Gewichtung angepasst und ein neuer Durchlauf gestartet. Dies wird wiederholt, bis das neuronale Netzwerk die gewünschten Werte ausgibt und dadurch das Ergebnis nicht verfälscht ist. (vgl. Trask, 2019)

1.2.2. Process Mining

Process Mining war ein Begriff, der vor gut 20 Jahren entstand, also rund um die Jahrtausendwende, und seitdem eine sehr große Relevanz aufgebaut hat. Einer der Forscher, welcher diese Technologie mitgeprägt hat, war Professor Will van der Aalst, der einen großen Beitrag daran hatte, dass diese Konzepte und Arbeitsweisen sich in der modernen Technik etabliert haben. In der modernen Zeit wird Process Mining verwendet, um aus verschiedensten Rohdaten neue und aufbereitete Informationen zu gewinnen, um sie anschließend zu analysieren und für den Menschen visuell darzustellen. Diese Methoden werden sehr oft für Geschäftsprozesse verwendet. Durch die gewonnenen neuen Informationen können diese visuell dargestellt werden und sind somit leichter zu identifizieren. Sollte es zu Abweichungen vom gewünschten Ergebnis kommen, kann man als Unternehmen eingreifen und somit das Geschäft optimieren. (vgl. Aalst, 2016)

Wie funktioniert nun diese Optimierung konkret? Grundlegend ist zu Process Mining zu sagen, dass in der Regel immer verschiedene Eventpunkte beziehungsweise Ereignisdaten mit unterschiedlichen Methoden überprüft, verändert oder angepasst werden. Durch diese Veränderung der Daten werden sie aufbereitet und ergeben ein Gesamtbild, aus dem später Schlüsse gezogen werden können. Diese Eventdaten folgen in der Regel einer zeitlichen Abfolge und können so identifiziert werden und

einer Zeitlinie zugeordnet werden. Aus dieser kann dann eine Visualisierung erfolgen. (vgl. Aalst, 2016)

Bei Process Mining gibt es verschiedene Miner, welche verwendet werden. Auf drei konkrete Anwendungsfälle wird nun kurz genauer eingegangen und erklärt, wie sie generell funktionieren (vgl. Process Mining Algorithms Simply Explained – Workfellow, o. J.):

Alpha Miner

Beim Alpha Miner werden die Eventlogs in eine zeitliche Sequenz geschalten, um ein Netz zu bilden. Diese Netze, auch Petri Netze genannt, zeigen die Daten und Entscheidungen in einer zeitlichen Abfolge und können durchlaufen werden. Durch diese Visualisierung ist es möglich, verschiedene Szenarien zu sehen und Entscheidungen durchzuspielen.

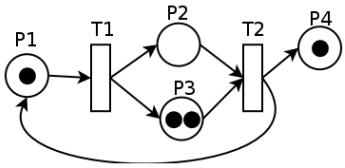


Abbildung 3: Alpha Miner

Heuristic Miner

Der Heuristic Miner funktioniert ähnlich wie der Alpha Miner, aber die erstellten Pfade werden mit Werten ausgestattet. Diese Werte repräsentieren die Häufigkeit der Aufrufe der Pfade. Durch diese Häufigkeit ist es leichter falsche beziehungsweise schlechte Pfade zu ermitteln und diese zu eliminieren.

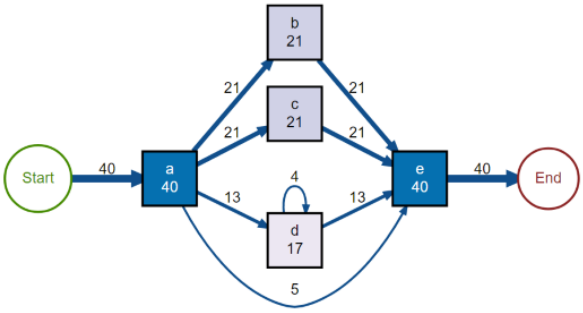


Abbildung 4: Heuristic Miner

Inductive Miner

Der Inductive Miner ist eine spezielle Form, da hier auch neue/mögliche Pfade erkannt und getestet werden. Es werden durch einen Algorithmus neue Knotenpunkte und Möglichkeiten iterativ entdeckt und getestet. Er bietet eine hohe Flexibilität und kann ein breites Spektrum abdecken.

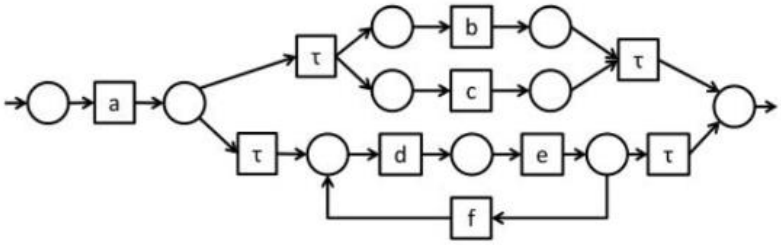


Abbildung 5: Inductive Miner

1.3. Zielsetzung

Das Ziel dieser Bachelorarbeit ist, eine Android-Applikation zu entwickeln, welche es ermöglicht, gegebene CSV-Dateien einzulesen und anschließend den Inhalt dieser CSV-Dateien grafisch und in Form von Listen für den User ersichtlich zu machen. Der Inhalt der CSV-Datei ist ein trainiertes Perceptron welches dargestellt und analysiert werden soll.

2. Entwurfsphase des Android Projektes

2.1. Applikationssystem

Bei der Wahl des Betriebssystems konnte zwischen IOS und Android frei gewählt werden. Die bestehende Applikation läuft auf IOS und Apple Geräten. Um eine neue Software zu entwerfen und nicht an der alten Version festzuhalten, wurde Android gewählt. Hier gibt es sehr viele Versionen und es wurde sich für „Android 8.1 Oreo“ entschieden, da diese eine Abdeckung von 90.2 % der am Markt vorhandenen Geräten bereitstellt.

Das komplette Projekt wurde in Android Studio entwickelt und getestet. Für das Testen wurde ein virtuelles Gerät verwendet, da kein Android-Gerät in realer Form zur Verfügung stand. Es wurde auf zwei verschiedenen Devices, namentlich „Pixel 2 XL API 30 R“ und „Pixel 2 API 27 Oreo“ getestet und programmiert.

2.2. Visuelle Darstellung Perceptron

Das Perceptron soll anhand der Daten, die zur Verfügung gestellt wurden, grafisch dargestellt werden. Hierzu wurde eine Recherche durchgeführt, welche Frameworks vergleicht, die für die spätere Implementierung verwendet werden sollen.

2.3. Android Mockup

Vor dem Start des Projektes wurde ein generelles Mockup erstellt, welches die einzelnen Aktivitäten darstellen und einen generellen Leitfaden geben sollte.

2.3.1. Startseite

Die Startseite sollte nach dem Laden der Applikation verfügbar und leer sein. Durch den Button „Import CSV“ sollte die Seite befüllt werden. Durch den Import der Daten wird eine Liste generiert, welche es dem User ermöglicht, die Daten aus der Datei einzusehen. Diese Liste sollte es möglich machen, durch einen Klick auf den einzelnen Knotenpunkt zu gelangen und genauere Informationen zu erhalten.

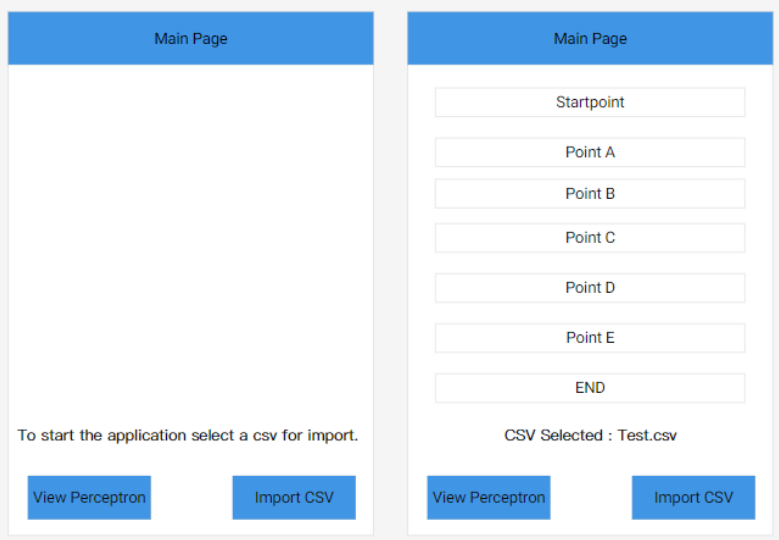


Abbildung 6: Mockup Startseite

2.3.2. Knotenpunkt

Der Knotenpunkt sollte einen realen Punkt am Perceptron darstellen und Hintergrundinformationen zu diesem Punkt geben. Diese Punkte sollten in der Reihenfolge dargestellt werden, wie sie eingelesen worden sind. Die Daten sollten anhand der gegebenen Informationen nach den höchsten und niedrigsten Werten sortiert werden.

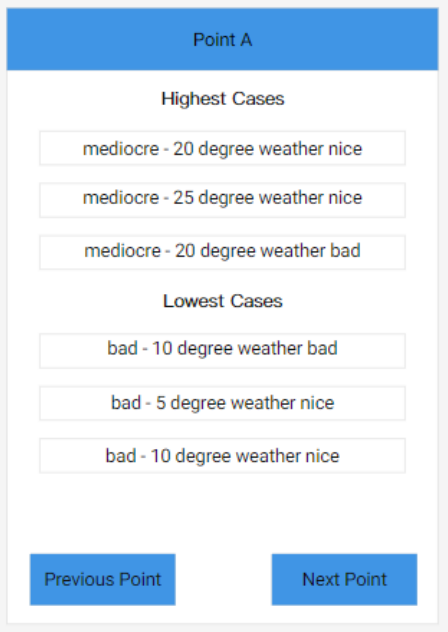


Abbildung 7: Mockup Knotenpunkt

2.3.3. Perceptron

Das Perceptron sollte anhand der eingelesenen Daten in einer Aktivität dargestellt und visualisiert werden. Die erste Überlegung war, diese Punkte mit Buttons zu verknüpfen, wurde aber in dem Entwurf schon verworfen.

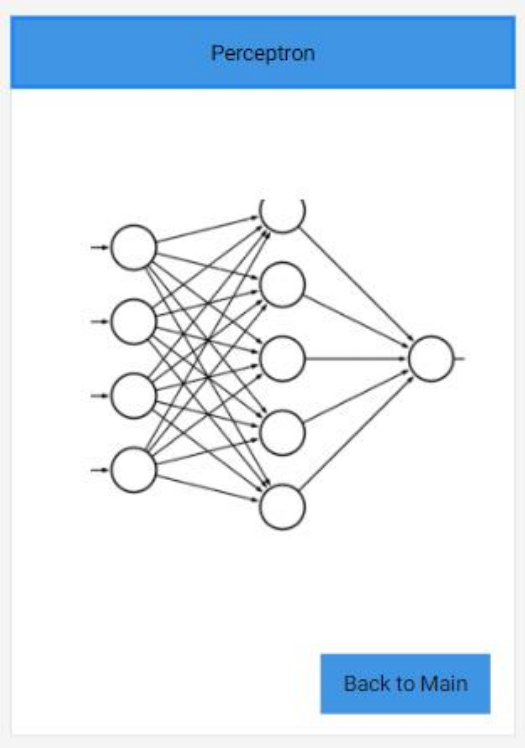


Abbildung 8: Mockup Perceptron

3. Implementierung und Ergebnisse

3.1. Generelle Ergebnisse

Die Voraussetzungen und Ziele der Bachelorarbeit konnten, bis auf die visuelle Darstellung des Perceptron komplett erfüllt werden. Auf dieses Problem wird später genauer eingegangen. Für den Rest der Applikation ist es möglich, eine CSV-Datei einzulesen und die Daten dann dynamisch in verschiedenen Aktivitäten einzusehen. Diese Aktivitäten werden als Node Punkte bezeichnet. Diese Nodes repräsentieren einen Schritt in dem Perceptron. Das folgende Kapitel soll die einzelnen Seiten und Funktionen darstellen und beschreiben.

3.2. Struktur des Projektes

Das Projekt hat zwei verschiedene Säulen. Die erste Säule ist der Java Code mit den einzelnen Aktivitäten und der Programmlogik. Die zweite Säule sind die einzelnen Grafik-Ressourcen für die Visualisierung und die Applikationskomponenten.

Für die Java Teile gibt es drei verschiedene Pakete:

Activities
Beschäftigt sich mit den einzelnen Aktivitäten und der Darstellung der Daten.

Adapter
Hier sind die einzelnen Adapter für die RecyclerViews der Aktivitäten implementiert. Diese beschäftigen sich mit der Befüllung der Listen mit den vorgegebenen Daten.

Javafiles
Hier ist das gesamte Backend mit den Funktionen und der Speicherung und Verarbeitung der Daten gesammelt.

Für die Ressourcen gibt es auch drei verschiedene wichtige Pakete:

Drawable
Beinhaltet alle design Aspekte der Buttons, Pfeile und anderwärtigen Grafik Aspekte.

Layout

Beinhaltet alle erstellten Designs der Aktivitäten, welche die generelle Struktur der Applikation bilden.

Raw

Hier werden die CSV-Dateien gespeichert, die vom System eingelesen werden und notwendig für das gesamte Projekt sind.

3.3. Struktur der Grunddaten

Für die Grunddaten des Projektes wurde die bereitgestellte CSV-Datei „df_perceptron.csv“ genommen und das Projekt wurde um diese Struktur herum gebaut. Um diese Daten und die einzelnen Werte für den Leser in reale Sprache zu übersetzen, wurde eine weitere CSV-Datei erstellt, welche es möglich macht, die Abkürzungen der einzelnen Events lesbar zu machen. Diese Datei hat die Bezeichnung „mapping.csv“ und wurde aus dem bereitgestellten Python Projekt extrahiert und die wichtigsten Informationen gebündelt dargestellt.

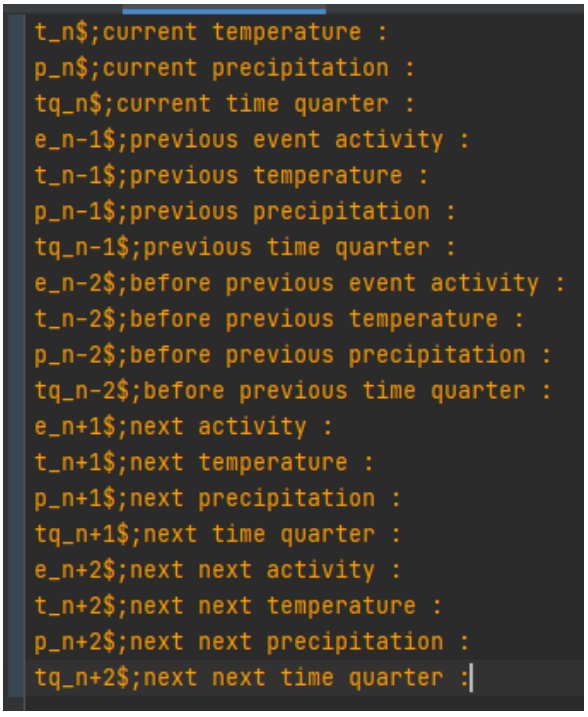


Abbildung 9: CSV Datei Eventinformation

Die CSV-Datei wird automatisch vom System eingelesen und verarbeitet. Diese Struktur ist essenziell für die Logik des Projektes und darf nicht vom User geändert werden. Um Änderungen an der Struktur durchzuführen, muss das Projekt generell und die Applikation angepasst werden.

3.4. Änderungen am Design

Generell ist zu den Änderungen anzumerken, dass das finale Design vom anfänglichen Mockup sehr weit entfernt ist. Die Buttons wurden in eine schönere Form gebracht und vereinheitlicht. Zusätzlich dazu wurden auch die einzelnen Knotenpunkte und Aktivitäten mit mehr Funktionen ausgestattet. Es kamen Möglichkeiten hinzu, einen Knotenpunkt genauer zu inspizieren und zu analysieren. Diese Änderungen sind ersichtlich im folgenden Kapitel.

3.5. Durchlauf durch die Applikation

Um eine generelle Übersicht über die Applikation zu geben, soll das folgende Kapitel die einzelnen Aktivitäten der Applikation zeigen und beschreiben.

3.5.1. Startseite

Das folgende Bild zeigt die fertige Startseite der Applikation. Hier kann durch den Button Import CSV eine CSV-Datei eingelesen werden. Dadurch wird ein Recycle View mit selektierbaren Buttons befüllt. Diese Buttons führen dann zu der nächsten Seite für einen bestimmten Knotenpunkt.

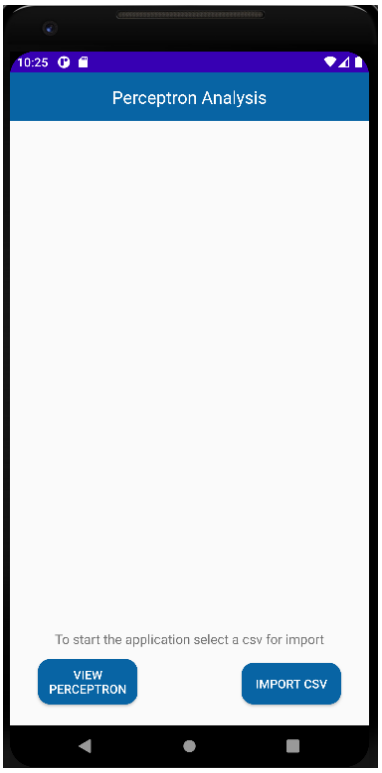


Abbildung 10: Applikation Startseite ungefüllt

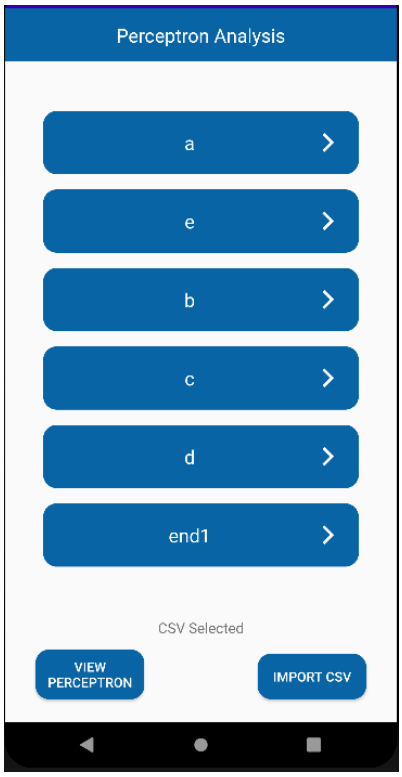


Abbildung 11: Applikation Startseite befüllt

3.5.2. Knotenpunktübersicht

Diese Knotenübersicht stellt einen Knoten des Perceptron dar, der aus der CSV-Datei importiert wurde. Diese Rohdaten wurden verändert und für den Benutzer in reale Sprache übersetzt. Hier gibt es zwei verschiedene Listen, welche die besten und schlechtesten Werte eines Knotenpunktes darstellen. Diese Listen sind dynamisch befüllt und die Größe kann auch verändert werden. Zusätzlich gibt es einen Button „Total Node Data“ welcher zu einer eigenen Aktivität mit den kompletten Datensätzen des Knotenpunktes führt. Am oberen Ende der Aktivität gibt es zwei Buttons, welche es möglich machen, zwischen den nächsten und vorherigen Knotenpunkten aus der Liste der CSV-Dateien zu wechseln und dort die Informationen zu den Knoten einzusehen.

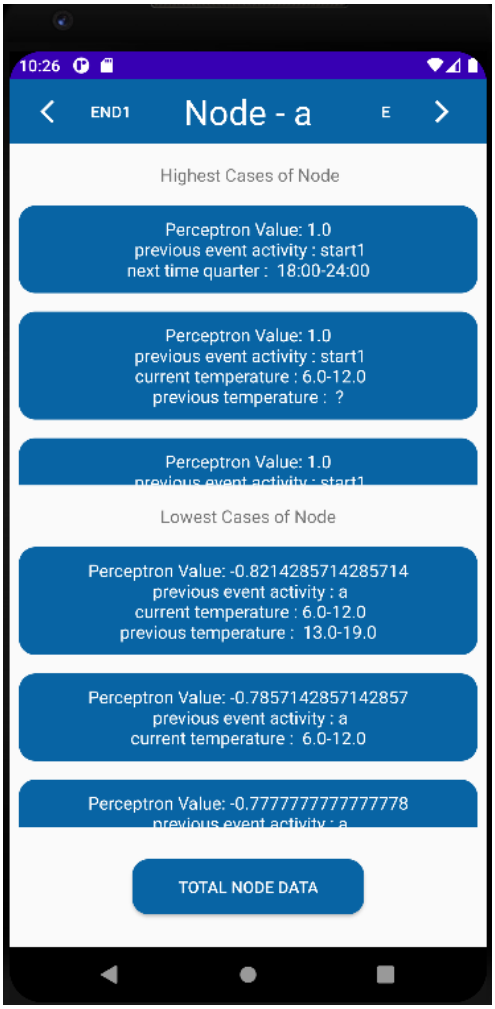


Abbildung 12: Applikation Knotenpunkt

3.5.3. Gesamtübersicht eines Knotens

Hier gibt es eine Gesamtübersicht über den Knoten und der User kann selbst selektieren, welche Informationen er beziehen will.

Es gibt vier Möglichkeiten: Gesamtdaten, Wetterdaten, Zeitdaten, Temperaturdaten

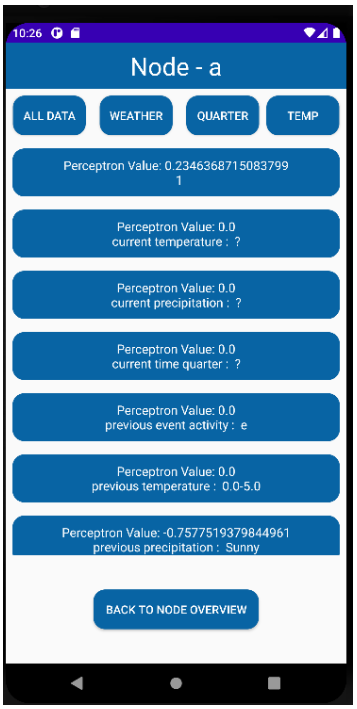


Abbildung 13: Applikation Knotenpunkt All Data

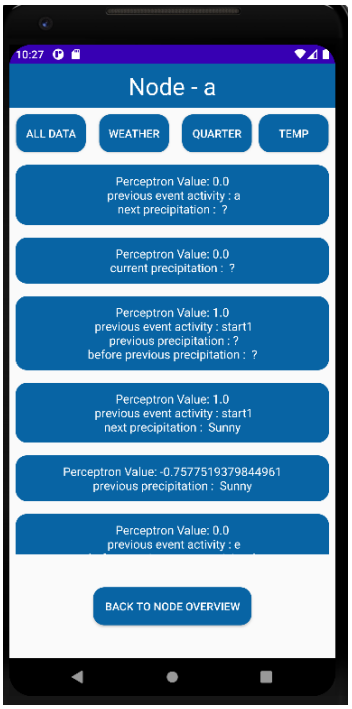


Abbildung 14: Applikation Knotenpunkt Weather

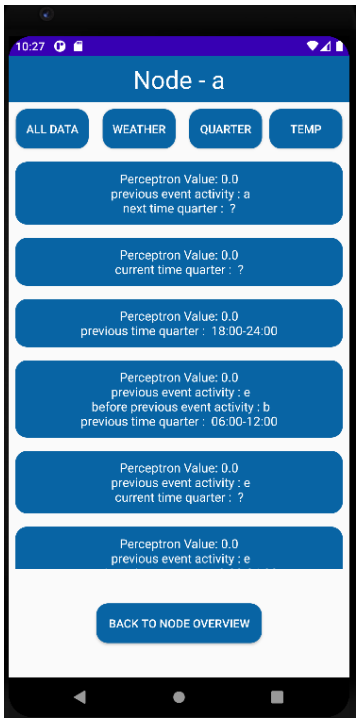


Abbildung 15: Applikation Knotenpunkt Quarter

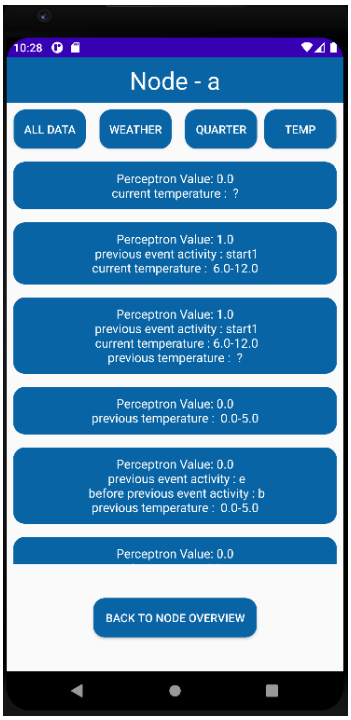


Abbildung 16: Applikation Knotenpunkt Temp

3.5.4. Perceptron-Visualisierung

Hier sollte das visualisierte Perceptron dargestellt werden. Diese Visualisierung wurde angedacht, aber es gibt kein Framework, welche diese zu einem gewünschten Maße darstellbar macht. Auf dieses Problem wird später noch genauer eingegangen. Die Aktivität wurde dennoch hinzugefügt, um für zukünftige Überarbeitungen des Projektes gleich eine Möglichkeit zu bieten, eine Visualisierung einzubinden.

Abbildung 17: Applikation Perceptron

4. „Lessons learned“ & Probleme bei der Implementierung

Durch das Programmieren, und die Bachelorarbeit an sich, traten verschiedene Probleme auf, welche behandelt werden mussten. Dadurch wurden neue Erkenntnisse und neues Wissen gewonnen. Diese werden im folgenden Kapitel behandelt und erläutert.

4.1. Leerstring am Anfang eines CSV-Files

Um die Übersetzungen in realer Sprache der einzelnen Events durchzuführen, wurde eine eigene CSV-Datei erstellt und die Daten ausgelesen. Nach dem Auslesen wurde bei der Darstellung der Daten eine Übersetzung durchgeführt. Hierfür gab es die Funktion „translate“, welche die einzelnen Werte aus der CSV-Datei einliest und dann

mit dem Wert vergleicht, der übersetzt werden soll und dann die reale Sprache ausgibt. Hier gab es aber einen Bug mit dem ersten Wert aus der CSV-Datei. Nach dem Einlesen des Strings aus der CSV hatte er immer einen versteckten extra Wert im Speicher. Dieser Wert hat es unmöglich gemacht, die beiden Werte mit .equals zu vergleichen, ob sie identisch sind. Auf dem Bild sieht man, dass der Count der beiden Werte unterschiedlich ist. Der linke Wert ist das Eingelesene und der rechte der zu übersetzende Wert. Dieses Problem konnte nicht behoben werden, außer durch einen „hardcodierten“ Vergleich der beiden Werte, welcher das Problem behoben hat, aber keine „schöne“ Lösung ist.

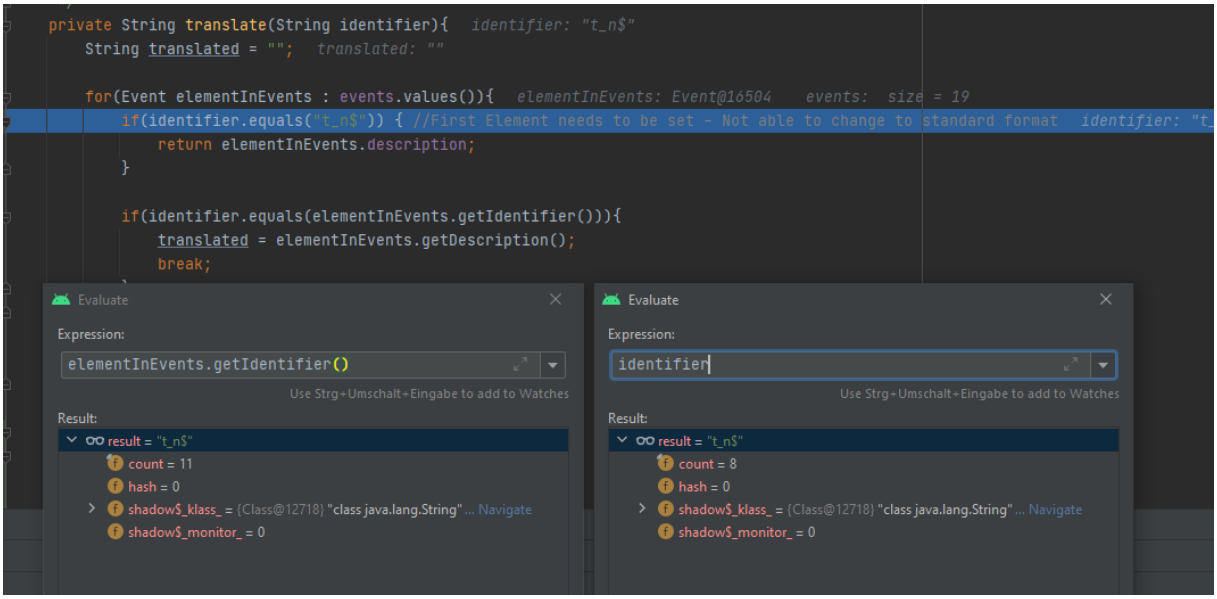


Abbildung 18: Applikation Leerstring Problem

4.2. Variabilität der Grunddaten

Die Grunddaten haben verschiedene Werte, die nicht immer genau der normalen Struktur folgen. Für diese Daten wurden Ausnahmen erstellt und eigene Fälle für die Übersetzung kreiert. Für diese Übersetzungen gab es Regelfälle, welche nach den Angaben des Professors geregelt wurden und auch fehlerhafte beziehungsweise nicht erwünschte Werte, welche extra behandelt werden mussten. Diese Einschränkungen der Daten führt dazu, dass das Projekt nicht komplett modular sein kann, da die Daten einem gewissen Standard unterliegen und nicht komplett modular gestaltet werden können.

Hier gab es 3 verschiedene Fälle:
Uhrzeiten abhängig vom Quartal des Tages, Temperaturen des Events, und Wetterbedingungen des Events.

4.2.1. Quartalsdaten

Für die Quartalsdaten gibt es 4 definierte Quartale des Tages, die auftreten können. Hierzu mussten die Werte übersetzt und für den User lesbar gemacht werden. Es gab aber einen Standardfall für alle Werte, die nicht dem Entsprechen. Diese Aufgabe hat die Funktion changeValueForQuarter durchgeführt. Hier wurden die Standarddaten durch die Uhrzeiten ersetzt und die falschen Werte wurden durch „?“ ersetzt.

```
private String changeValueForQuarter(String value){
    switch(value) {
        case "q1":
            return "00:00-06:00";
        case "q2":
            return "06:00-12:00";
        case "q3":
            return "12:00-18:00";
        case "q4":
            return  "18:00-24:00";
        default:
            return "?";
    }
}
```

Abbildung 19: Applikation Quartalsdaten

4.2.2. Temperaturdaten

Bei den Temperaturdaten gibt es auch, wie bei den Quartalsdaten, einen Standardfall, der eintritt, wenn eine Temperatur im Format „00:00-00:00“ ist. Die Nullen sind im Standardfall die richtigen Temperaturen der Daten. Hier gibt es aber auch mehrere extra Fälle, die durch das Trainieren des Perceptron auftreten. Ein Beispiel wäre der Wert „-91.0“. Dieser Wert ist keine Temperatur, sondern tritt nur auf, wenn das Perceptron trainiert wird und sollte durch ein "?" ersetzt werden. Hierzu wurde eine leichte Methode erstellt, nämlich, dass die Länge des Strings immer mindestens 6 sein muss um die Struktur „00:00-00:00“ zu gewährleisten.

```
private String changeForTemperature(String value) {
    return value.length() <= 6 ?
            "?"
            :
            value;
}
```

Abbildung 20: Applikation Temperaturdaten

4.2.3. Wetterbedingungen

Auch bei den Wetterdaten gibt es einen Standardfall. Der Wert „0.0“ beschreibt das Wetter als „sonnig“ und der Wert „1.0“ beschreibt das Wetter als regnerisch. Hier kommt es aber auch zu anderen Werten, die nicht in dieses Spektrum passen. Genau wie bei den Temperaturen ist dies für die Darstellung der Daten notwendig. Hier wurde auch eine extra Funktion gebaut, die es möglich macht, diese Standardfälle zu filtern und Fehlerfälle gegen „?“ zu tauschen.

```
private String changeForPrecip(String value){
    switch(value){
        case "0.0":
            return "Sunny";
        case "1.0":
            return "Rainy";
        default:
            return "?";
    }
}
```

Abbildung 21: Applikation Wetterdaten

4.3. Perceptron Darstellung

Die Perceptron Darstellung war vom Professor als Ziel gewünscht, konnte aber nicht erfüllt werden. Es wurden etliche Recherchen durchgeführt, um ein Framework zu finden, welches es ermöglicht, das trainierte Perceptron darzustellen. Es gibt zwar für Java verschiedene Frameworks, die es ermöglich, Perceptron in Graphen darzustellen, aber eine wirkliche Darstellung mit Punkten und visuellen Pfeilen gibt es nicht auf dem aktuellen Markt. Diese Graphen sind klassische Linien- oder Balkengraphen, die den Zweck nicht erfüllen würden. Um eine andere Methode auszuprobieren, wurde versucht mittels Canvas die Punkte darzustellen und die notwendigen Pfeile der Verbindungen der einzelnen Knoten zu zeichnen. Diese Grafiken waren aber für den kleinen Bildschirm des Telefons nicht geeignet und wurden aus diesen Gründen weggelassen. Zusätzlich zu der Lesbarkeit wurde auch die Komplexität mit mehreren Knoten viel zu unübersichtlich und nicht anwendbar.

5. Resümee

Die Bachelorarbeit war eine gute Gelegenheit, neue Informationen und Arbeitsweisen im Bereich von Android zu entdecken. Die größten Herausforderungen waren die anfänglichen Nachforschungen über die Struktur und Funktionsweisen des Perceptron und der Analyse der Grunddaten. Zusätzlich dazu gab es mehrere Probleme bei der Modularität des Projektes und der Applikation, da die Daten sehr stark definiert sind und nicht viel Änderungen zulassen. Das Projekt, war bis auf die visuelle Darstellung des Perceptron ein Erfolg und hat eine neue Begeisterung für Android und Appentwicklung im Generellen geweckt.

Literaturverzeichnis

Aalst, W. M. P. van der. (2016). Process Mining: Data Science in Action. Springer.

Process Mining Algorithms Simply Explained – Workfellow. (o. J.). Abgerufen 3. August 2023, von https://www.workfellow.ai/learn/process-mining-algorithms-simply-explained

Sabry, F. (2023). Perceptrons: Fundamentals and Applications for The Neural Building Block. One Billion Knowledgeable.

Trask, A. W. (2019). Neuronale Netze und Deep Learning kapieren: Der einfache Praxiseinstieg mit Beispielen in Python. MITP-Verlags GmbH & Co. KG.